第 3 章　选择结构程序设计

在顺序结构程序中，所有语句按照自上而下的顺序执行，即按照代码书写的先后顺序，执行完上一条语句后执行下一条语句，直到程序结束。这是最常见也是最简单的程序结构。然而在实际问题中，往往需要根据不同的条件或情况来选择执行不同的操作任务，这时顺序结构就无法完成这些任务。这就需要用到本章将要学习的选择结构。

在 C 语言中，常用的选择结构语句有两种。

（1）if 语句：用来实现两个分支的选择结构。

（2）switch 语句：用来实现多分支的选择结构。

在学习本章知识时，读者要体会选择结构程序中蕴含的道理：人生道路上存在很多选择，做每一个决定之前一定要经过深思熟虑，因此树立正确的人生观和价值观将有利于做出更好的选择。

3.1　选 择 引 例

【例 3-1】 中国有句俗话：“三天打鱼两天晒网”。编程实现，当每年的 1 月 1 日为第一天时，输入天数，判断今天是打鱼还是晒网。

编程提示：以 1 月 1 日为第一天，每 3 天打鱼，每 2 天晒网，打鱼和晒网每 5 天为一个周期，因此计算天数对 5 求余数，判断结果与 3 的大小，以确定今天是打鱼还是晒网。

```
#include <stdio.h>
int main()
{
    int n;
    printf("输入天数：");
    scanf("%d",&n);                          /*输入天数*/
    if (n%5==0||n%5==4)                      /*判断是打鱼还是晒网*/
        printf("今天晒网！\n");
    else
        printf("今天打鱼！\n");
    return 0;
}
```

程序运行结果如下：

```
输入天数：10↙
今天晒网！
```

做人必须要脚踏实地，切不可三天打鱼两天晒网。要想成为一个德才兼备的人，就必须全面提高自己的各方面能力，要做到长期坚持，虽然过程会比较辛苦，但最后的结果往

往会出乎意料。

【同步练习】 输入一个正整数，判断它的奇偶性。

提示：判断一个数的奇偶性，即判断该数是否能被 2 整除。

3.2 选 择 条 件

在实际生活中，有些问题需要根据某个条件进行判断和选择后才决定完成哪些任务。这里的条件抽象出来后，在程序中，通常被描述为关系表达式和逻辑表达式。

3.2.1 关系运算符与关系表达式

比较或判断两个数值是否符合给定条件时，需要使用关系运算符。例如，一个正整数是否能被 2 整除；两个变量 *a* 和 *b* 比较大小；构成三角形的条件是任意两条边的和大于第三边，这些都需要利用关系运算符与关系表达式。

1. 关系运算符

C 语言提供了 6 种关系运算符，如表 3-1 所示。

表 3-1 关系运算符

运 算 符	含 义	目	使 用	优 先 级	结 合 性
<	小于比较	双目	exp1 < exp2	高	自左向右
<=	小于或等于比较	双目	exp1 < = exp2		自左向右
>	大于比较	双目	exp1 > exp2		自左向右
>=	大于或等于比较	双目	exp1 >= exp2		自左向右
==	相等比较	双目	exp1 = = exp2	低	自左向右
!=	不相等比较	双目	exp1 ! = exp2		自左向右

2. 关系表达式

用关系运算符将两个数值或数值表达式连接起来的式子，称为关系表达式。例如，a>b、'a'>'b'、a+b>c、（a>b）>（b>c）。关系表达式的值是一个逻辑值，即“真”或“假”。在 C 语言的逻辑计算中，以“1”代表“真”，以“0”代表“假”。

3.2.2 逻辑运算符与逻辑表达式

若判断的一个关系不是一个简单条件，而是由几个给定简单条件组合的复合条件。例如，假定三角形的 3 条边分别为变量 *a*、*b*、*c*，判断这 3 条边是否可以构成一个三角形。由于构成三角形的条件是任意两条边的和大于第三边，即 *a*+ *b* >*c*、*b* + *c* >*a*、*c* +*a*> *b* 同时成立，所以就需要用到逻辑运算符，即（*a*+ *b* > *c*）&&（*b* + *c* >*a*）&&（*c* +*a*> *b*）。

1. 逻辑运算符

C 语言提供了 3 种逻辑运算符，如表 3-2 所示。

2. 逻辑表达式

逻辑表达式就是用逻辑运算符将关系表达式或其他逻辑量连接起来的式子。逻辑表达式的值是一个逻辑值，即“真”或“假”。在 C 语言中表示逻辑运算结果时，以“1”代表“真”，

以“0”代表“假”，但在判断一个量时，非“0”的值为真，“0”代表“假”。逻辑运算的真值表如表 3-3 所示。

表 3-2　逻辑运算符

运　算　符	含　　义	目	使　　用	优　先　级	结　合　性
!	逻辑非	单目	!exp	单目高于所有双目	自右向左
&&	逻辑与	双目	exp1&& exp2	高	自左向右
||	逻辑或	双目	exp1 || exp2	低	自左向右

表 3-3　逻辑运算的真值表

a	***b***	**!*a***	**!*b***	***a* &&*b***	***a* || *b***
非 0	非 0	0	0	1	1
非 0	0	0	1	0	1
0	非 0	1	0	0	1
0	0	1	1	0	0

逻辑非：这种逻辑运算最简单，即真变假；假变真。

逻辑与：当条件同时为真时，结果才为真；其他情况都为假。

逻辑或：只要有一个条件为真，结果就为真；只有条件同时为假时，结果才为假。

例如，判断某年是否为闰年，可以用逻辑表达式来表示。

闰年的条件：

（1）能被 4 整除，但不能被 100 整除。

（2）能被 400 整除。

如果满足两个条件之一，就可以判断为闰年。

逻辑表达式如下：(year%4==0 && year%100!=0) || year%400==0。

注意：逻辑与&&和逻辑或||具有短路特性。

（1）*a*&&*b*，若 *a* 为真，则 *b* 需要判断；若 *a* 为假，*b* 不必判断。

（2）*a* ||*b*，若 *a* 为真，则 *b* 不必判断；若 *a* 为假，则 *b* 需要判断。假定 *a* =1，*b*=0，执行（*a* =2>3）&&(*b*=4>1)，会发现输出 *a* 为 0，*b* 为 0；这就说明当执行 *a* =2>3 时，逻辑与&&左边已经为假，逻辑与&&右边的表达式 *b*=4>1 根本没有执行。

3.3　if　语　句

C 语言中的 if 语句主要用来实现两个分支的选择。if 语句的结构比较灵活，既可以实现简单的选择，也可以实现多分支的选择。

3.3.1　if 语句的一般形式

【例 3-2】 编程实现，比较两个实数的大小，输出较大者。

编程提示：由于需要比较两个数的大小，因此要用选择结构。两个数的比较有 3 种情况：大于、等于或小于，这时要用到关系运算符。该选择条件描述如下：关系表达式 *a*>=*b*。

方法 1：

```
#include <stdio.h>
int main()
{
    float a,b;
    printf("输入两个实数(中间以空格间隔):");
    scanf("%f%f",&a,&b);     /*输入两个实数, 中间以空格间隔*/
    if (a>=b)                /*a 大于或等于 b*/
       printf("两个数中的较大者为%5.2f\n",a);
    else                     /*a 小于 b*/
       printf("两个数中的较大者为%5.2f\n",b);
    return 0;
}
```

程序运行结果如下：

```
输入两个实数(中间以空格间隔): 23 21↙
两个数中的较大者为 23.00
```

方法 2：

```
…
if (a>=b)                    /*a 大于或等于 b*/
    max=a;
else                         /*a 小于 b*/
    max=b;
printf("两个数中的较大者为%5.2f\n",max); /*将较大者赋值为 max*/
…
```

if 语句一般形式如下：

```
if ( 表达式 )   /*if…else…形式,实现双分支*/
    语句 1;
[else
    语句 2; ]
```

if 语句的执行过程是，如果条件表达式的值为真，则执行语句 1；否则执行语句 2。执行流程如图 3-1 所示。

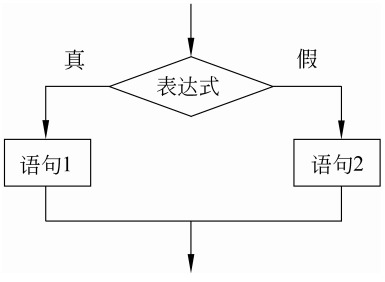

图 3-1　if 语句的一般形式流程图

使用 if 语句要注意以下几点。

（1）if 语句后的“表达式”可以是关系表达式、逻辑表达式或数值表达式。不论是哪种表达式，其结果都为“真”或“假”。

（2）语句 1 和语句 2 可以是一个简单的语句（赋值语句或输入输出语句），也可以是一个复合语句（作为一个整体执行），还可以是另一个 if 语句（即内嵌一个或多个 if 语句，if 语句的嵌套）。

（3）else 子句用“[]”括起来表示可选。但 else 子句不能单独使用，必须和 if 语句配对使用。

【例 3-3】 编程实现，输入 3 条边，判断能否构成一个三角形，若能构成一个三角形，则计算其面积；否则输出“这 3 条边不能构成一个三角形!”。

编程提示：判断输入的 3 条边能否构成三角形，需要用到选择结构。构成三角形的条件是任意两条边的和大于第三边，即逻辑表达式(*a*+*b*>*c*) &&(*a*+ *c* > *b*) && (*b* + *c* >*a*)。

用海伦公式求三角形面积 *S*：

$$S=\sqrt{s(s-a)(s-b)(s-c)}\text{，其中 } s=\frac{(a+b+c)}{2}$$

将上述两个数学公式转换为 C 语言的表达式，即 sqrt(s*(s−a)*(s−b)*(s−c))和 s=(a+b+c)/2.0。

```
#include <stdio.h>
#include <math.h>
int main()
{
    double a,b,c;
    printf("input a,b,c(以","间隔：");
    scanf("%lf,%lf,%lf",&a,&b,&c);  /*输入三角形的 3 条边长度，中间以","间隔*/
    if (a+b>c && a+c>b && b+c>a)    /*判断 3 条边是否能构成三角形*/
    {
        double area,s;
        s=(a+b+c)/2.0;
        area=sqrt(s*(s-a)*(s-b)*(s-c));  /*海伦公式计算三角形面积*/
        printf("area=%.2f\n", area);
    }
   else
        printf("这 3 条边不能构成一个三角形!\n");
    return 0;
}
```

程序运行结果如下：

```
input a,b,c(以逗号间隔)：3,4,5↙
area=6.00
```

说明：sqrt(x)是库函数，其功能是求 x 的平方根，要求 x 必须大于或等于 0。在调用此函数时，需要在 main()函数前加#include <math.h>。

【同步练习】 比较 3 个实数的大小，输出较大者。

提示：求 3 个实数 a、b、c 中的较大者。假定 a 是 3 个数中的较大者，并赋值给 max，然后 max 和 b 比较，若 b 大于 max，则将 b 赋值给 max；再 max 和 c 比较，若 c 大于 max，则将 c 赋值给 max，最后输出 max。这种方法被称为“打擂台”法。

3.3.2 用 if 语句实现简单的选择结构

【例 3-4】 编程实现，输入两个整数，按由小到大的顺序输出这两个数。

编程提示：排序的本质就是两个数大小的比较，按由小到大的顺序输出 *a* 和 *b* 时，需要比较 *a* 和 *b*，当 *a* 大于 *b* 时，将 *a* 和 *b* 进行交换后，输出 *a* 和 *b*；当 *a* 小于或等于 *b* 时，则不做任何操作，直接输出 *a* 和 *b*。

两个变量的值相互交换时，不能直接相互复制。

```
{
   a=b;     /*把 b 的值赋给 a, a 的值等于 b 的值*/
   b=a;     /*再把 a 的值赋给 b, b 的值没有改变*/
}
```

这就像是要实现一瓶牛奶和一瓶可乐互换，需要借助一个空的瓶子才行；试图通过两个瓶子直接倒来倒去的方法是无法实现牛奶和可乐互换的。

正确实现两个变量的值相互交换的方法，需要借助一个新变量。

```
{           /*将 a 和 b 的值互换*/
   t=a;
   a=b;
   b=t;
}
```

完整的程序代码如下：

```
#include <stdio.h>
int main()
{
   int a,b,t;
   printf("输入两个整数 a、b：");
   scanf("%d%d",&a,&b);     /*输入两个整数*/
   if (a>b)                 /*当 a 大于 b 时,执行下面的操作;否则,不做其他操作*/
   {                        /*将 a 和 b 的值互换*/
      t=a;
      a=b;
      b=t;
   }
   printf("两个整数按由小到大的顺序输出：%d,%d \n",a,b);
   return 0;
}
```

程序运行结果如下：

```
输入两个整数 a、b：8 5↙
两个整数按由小到大的顺序输出：5,8
```

if 不带 else 子句的形式如下：

```
if ( 表达式 )
    语句;
```

if 语句的执行过程是，如果条件表达式的值为真，则执行语句；否则不做任何操作，直接执行 if 结构下面的语句。

执行流程如图 3-2 所示。

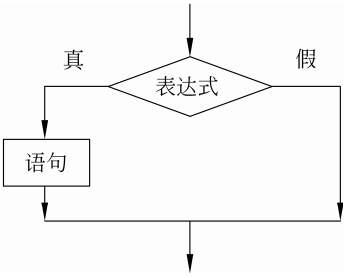

图 3-2　if 语句的一般形式流程图

【例 3-5】 有一个函数：

$$y=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$$

用 if 语句编程实现，当输入一个 x 的值后，输出相应的 y 值。

编程提示：这是一个分段函数，x 的 3 个不同域，y 的值分别对应为－1、0、1。可以把每个域看作一个简单的选择，因此，可以使用 3 个独立的 if 语句。

```
#include <stdio.h>
int main()
{
    int x,y;
    scanf("%d",&x);
    if (x>0)   y=1;
    if (x==0)  y=0;
    if (x<0)   y=-1;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```

程序运行结果如下：

```
3↙
x=3,y=1
```

【同步练习】 输入 3 个整数并按由小到大的顺序输出。

提示：*a*、*b*、*c* 3 个数排序，先 *a* 与 *b* 比较，使 *a* 小于 *b*；然后 *a* 与 *c* 比较，使 *a* 小于 *c*；最后 *b* 与 *c* 比较，使 *b* 小于 *c*。注意比较顺序。

3.3.3 用 if 语句实现多分支选择结构

if 语句不仅可以实现单分支和双分支的选择结构，而且可以实现多分支的选择结构。通常情况下，if 语句实现多分支的形式有两种。

【例 3-6】 有一个函数：

$$y=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$$

用 if…else 语句编程实现，当输入一个 *x* 的值后，输出相应的 *y* 值。

编程提示：这是一个分段函数，每个域对应一个值，且相邻的两个域之间存在双分支选择，因此可以使用 if 语句实现多分支的选择结构。

第 1 种，称为 if 语句的第 3 种形式，即在 else 分支上嵌套 if 语句。

```
#include <stdio.h>
int main()
{
   int x,y;
   scanf("%d",&x);
   if (x>0)                          /*if 语句的第 3 种形式*/
      y=1;
   else if (x==0)
      y=0;
   else
      y=-1;
   printf("x=%d,y=%d\n",x,y);
   return 0;
}
```

它的一般形式如下：

```
if ( 表达式 1 )
     语句 1;
else if ( 表达式 2 )
          语句 2;
   else if ( 表达式 3 )
        语句 3;
        …
        else if (表达式 n)
                 语句 n;
             else
                 语句 n+1;
```

该语句的执行过程是：若表达式 1 的值为真，则执行语句 1；否则，若表达式 2 的值为真，则执行语句 2；其余以此类推，若前 *n* 个表达式都为假，则执行语句 *n*+1。其中任何一个 else if 子句都表示否定上一个 if 后面的表达式，同时满足自己 if 后面的表达式。注意，else 与 if 中间至少有一个空格。

执行流程如图 3-3 所示。

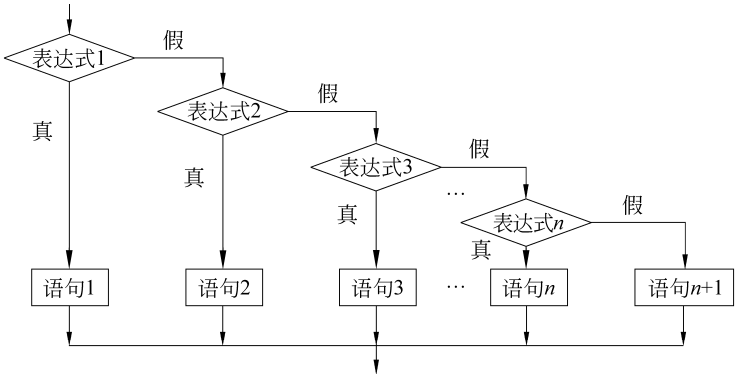

图 3-3　if 语句第 3 种形式流程图

第 2 种，if 嵌套语句，即在 if…else 分支上嵌套 if 语句。

```
#include <stdio.h>
int main()
{
    int x,y;
    scanf("%d",&x);
    if (x>0)
        y=1;
    else
        if (x==0)                          /*内嵌的 if*/
            y=0;
        else
            y=-1;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```

它的一般形式如下：

```
if ( 表达式 1 )   /*if 嵌套形式，实现多分支*/
    if ( 表达式 2 )
        语句 1;          } 内嵌的 if
    else
        语句 2;
else
```

```
    if ( 表达式3 )
        语句3;
    else
        语句4;
```
内嵌的 if

注意：由于 else 不能单独成句，因此当 if 语句进行嵌套时，应注意 else 的匹配问题，即 else 总是与离它最近的未配对的 if 配对。

根据算法的不同，if 语句的嵌套形式不是唯一的，建议将内嵌的 if 语句放在外层 if…else 语句的 else 子句中。这种情况下，if 语句嵌套形式实际上完全等价于 if 语句的第 3 种形式。重要的是，它避免了 else 与 if 语句进行匹配时引起的程序歧义。

从形式上看，if 语句的第 3 种形式明显比 if 嵌套语句更加直观和简洁，因此当使用 if 语句实现多分支结构时，建议使用前者。

【例 3-7】 编程实现，将一个输入的百分制成绩转换为对应的五级制成绩，即 90～100 分的成绩等级为 A，80～89 分的成绩等级为 B，70～79 分的成绩等级为 C，60～69 分的成绩等级为 D，0～59 分的成绩等级为 E。若输入的值不在 0～100，则程序输出“这是一个非法数据!”。

编程提示：输入一个 0～100 的数，当它为 90～100，即(score>=90) && (score<=100)时，等级为 A；当它为 80～90，即(score>=80) && (score<90)时，等级为 B；……当它为 0～60，即(score>=0) && (score<60)时，等级为 E。

第 1 种（独立的 if 语句）：

```
#include <stdio.h>
int main()
{
   int score;
   printf("请输入一个百分制成绩：");
   scanf("%d",&score);                    /*输入一个百分制的成绩*/
   if (score<0||score>100)
      printf("这是一个非法数据! \n");
   if (score<=100 && score>=90 )          /*90分以上*/
      printf(" %d --> A \n",score);
   if (score<90&& score>=80)              /*80～89分*/
      printf(" %d --> B \n",score);
   if (score<80 && score>=70)             /*70～79分*/
      printf(" %d --> C \n",score);
   if (score<70 && score>=60)             /*60～69分*/
      printf(" %d --> D \n",score);
   if (score<60 && score>=0)              /*60分以下*/
      printf(" %d --> E \n",score);
   return 0;
}
```

程序运行结果如下：

```
请输入一个百分制成绩：85↙
```

```
85 --> B
```

第 2 种（if 语句的第 3 种形式）：

```
…
if (score<0||score>100)
   printf("这是一个非法数据! \n");
else if (score>=90)                                  /*90 分以上*/
       printf(" %d --> A \n",score);
   else if (score>=80)                               /*80～89 分*/
       printf(" %d --> B \n",score);
       else if (score>=70)                           /*70～79 分*/
          printf(" %d --> C \n",score);
          else if (score>=60)                        /*60～69 分*/
             printf(" %d --> D \n",score);
                else                                 /*60 分以下*/
                   printf(" %d --> E \n",score);
…
```

注意：else 子句表示否定所对应的 if 子句的表达式，因此 else if 后的表达式不必再包含否定上一个 if 子句的表达式。例如：

```
if (score>=90)
   语句 1;
else if (score>=80&&score<90)      /* 此处不必给出表达式 score<90 */
   语句 2;
```

第 3 种（if 嵌套语句）：

```
…
if (score<0||score>100)
   printf("这是一个非法数据! \n");
else
   if (score>=90)                                    /*90 分以上*/
      printf(" %d --> A \n",score);
   else
      if (score>=80)                                 /*80～89 分*/
         printf(" %d --> B \n",score);
      else
         if (score>=70)                              /*70～79 分*/
            printf(" %d --> C \n",score);
         else
            if (score>=60)                           /*60～69 分*/
               printf(" %d --> D \n",score);
            else                                     /*60 分以下*/
               printf(" %d --> E \n",score);
      …
```

注意：else 子句表示否定所对应的 if 子句的表达式，因此 else 中的内嵌 if 后的表达式不必再包含否定上一个 if 子句的表达式。例如：

```
if (score>=90)
    语句1;
else
if (score>=80&&score<90)            /*此处不必给出表达式score<90*/
    语句2;
```

3 种方法均可以实现编程，但由于分支较多且各条件之间有逻辑关系，因此使用包含 else 子句的 if 更简洁；而 if 的第 3 种形式与 if 嵌套形式相比较，前者更直观。因此，解决同类问题时建议使用 if 的第 3 种形式。

【同步练习】 输入一个字符，判断该字符是大写英文字母、小写英文字母、数字字符、空格还是其他字符。

提示：多种情况的判断，采用选择分支结构中的 if 的第 3 种形式。

3.4 switch 语句

C 语言中 switch 语句也可以实现多分支的选择结构，但由于它的结构特点，即只能对整型或字符型数据进行匹配，因此使用时没有 if 语句更具有广泛性。

3.4.1 switch 语句的一般形式

【例 3-8】 编程实现，将一个输入的五级制成绩转换为对应的百分制分数段，即 A 等为 90～100 分，B 等为 80～89 分，C 等为 70～79 分，D 等为 60～69 分，E 等为 60 分以下。

编程提示：将学生的五级制成绩显示为对应的百分制分数段，很明显这是一个多分支选择问题。使用 switch 语句实现多分支选择结构。

```
#include <stdio.h>
int main()
{
    char grade;
    printf("请输入一个五级制成绩(如A、B、C、D、E)：");
    scanf("%c",&grade);                          /*输入一个字符*/
    printf(" %c 对应的百分制分数段为",grade);
    switch(grade)                                /*grade为字符型*/
    {
        case 'A': printf("90～100\n");break;
        case 'B': printf("80～89\n");break;
        case 'C': printf("70～79\n");break;
        case 'D': printf("60～69\n");break;
        case 'E': printf("60分以下\n");break;
        default:  printf("输入数据有误!\n");break;
        /*字符不是A、B、C、D、E时，执行此语句 */
    }
```

```
    return 0;
}
```

程序运行结果如下：

```
请输入一个五级制成绩(如A、B、C、D、E)：B↙
B对应的百分制分数段为80～89
```

switch 语句的一般形式如下：

```
switch (表达式)
{
   case 常量1:  语句1; break;
   case 常量2:  语句2; break;
   …
   case 常量n:  语句n; break;
    [default:   语句m;]
}
```

switch 语句的执行过程是，首先计算 switch 后面的表达式的值，然后将该值依次与各 case 子句后的常量进行比较，当它们相等时，执行相应的 case 子句，当执行到 break 语句时，程序直接跳转到 switch 语句后面的语句，表示 switch 语句执行完毕。若没有与 switch 后面的表达式的值相等的 case 常量，程序执行 default 子句。

使用 switch 语句要注意以下几点。

（1）switch 后面“( )”中的“表达式”只能是整型或字符型数据。

（2）switch 下面的“{ }”内是一个复合语句，这个复合语句包含多个 case 子句和最多一个 default 语句（根据程序要求，也可以没有 default 语句）。

（3）case 后面的常量（或常量表达式）与 switch 后面“( )”中的“表达式”类型一致。注意，case 与常量中间至少有一个空格且常量后面是“:”。每个 case 常量必须互不相同；但多个 case 常量可以执行同一语句。例如：

```
⋮
case 'A':
case 'B':
case 'C':
case 'D': printf(">60\n");break;
⋮
```

（4）在 case 子句中虽然包含了一个以上的子句，但可以不必用“{ }”括起来，程序会自动顺序执行本 case 后面所有的语句。每个 case 子句中都包含一个 break 语句，但最后一条语句不论是 case 子句还是 default 子句，都可以没有 break 语句。

（5）每个 case 子句中必须有 break 语句，否则程序将顺序执行下面所有 case 子句，直到遇到 switch 的“}”为止。这样，switch 语句将无法实现多分支选择结构。

【同步练习】 编写程序实现以下功能，当输入 A 或 a 时，屏幕显示“打开文件!”；当输入 B 或 b 时，屏幕显示“关闭文件!”。

提示：多分支结构，可以使用 switch 语句。其中 case 'A'和 case 'a'子句执行同一条语句；case 'B'和 case 'b'子句执行同一条语句。

3.4.2 用 switch 语句实现多分支选择结构

要想使用 switch 语句解决多分支问题，首先要看其条件是否能够转换为 switch 关键字后面“表示式”要求的整型或字符型数据。因此，不是所有的多分支结构都可以用 switch 语句实现。

【例 3-9】编程实现，输入一个百分制成绩，编程将它转换为对应的五级制成绩，即 90～100 分的成绩等级为 A，80～89 分的成绩等级为 B，70～79 分的成绩等级为 C，60～69 分的成绩等级为 D，0～59 分的成绩等级为 E。若输入的值不为 0～100，则程序输出“这是一个非法数据!”。

编程提示：本题在 3.4.1 节中已经分析过，是一个典型的多分支选择结构，因此可以使用 switch 语句解决。但由于 switch 后面的“表达式”要求是整型数据，题目描述中又没有直接给出，这就需要找出将题目中区间值转换为整型数据的表达式。经过分析发现，百分制的成绩对 10 取整后，得到的数字恰好可以表示某个分数段，即

100：　十位为 10　100/10=10；
90～99：　十位为 9　90/10～99/10=9；
80～89：　十位为 8　80/10～89/10=8；
70～79：　十位为 7　70/10～79/10=7；
60～69：　十位为 6　60/10～69/10=6；
……

因此，定义整型变量 score，switch 后面的“表达式”即为 score/10。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int score;
   char grade;
   printf("请输入一个百分制成绩：");
   scanf("%d",&score);                    /*输入一个百分制的成绩*/
   if (score<0||score>100)
   {  printf("这是一个非法数据! \n ");
      exit(0);
   }
   else
   {
      switch(score/10)                    /*将区间值转换为整型数据*/
      {
         case 10:
         case 9: grade='A';break;         /*90 分以上*/
         case 8: grade='B';break;         /*80～89 分*/
         case 7: grade='C';break;
```

```
            case 6: grade='D';break;
            case 5:                        /*60 分以下*/
            case 4:
            case 3:
            case 2:
            case 1:
            case 0: grade='E';
        }
        printf(" %d --> %c\n ",score,grade);
    }
    return 0;
}
```

程序运行结果如下：

```
请输入一个百分制成绩：97↙
97 --> A
```

说明：函数 exit()的作用是终止整个程序的执行，强制返回操作系统，并将 int 型参数 code 的值传给调用进程（一般为操作系统）。当 code 为 0 时，便是程序正常退出。调用函数 exit()时，需要在程序开头加#include<stdlib.h>。

当输入分数小于 60 时，case 5、case 4、case 3、case 2、case 1 和 case 0 都执行同一条语句，即五级成绩都为 E。根据 switch 语句的一般形式，程序可以修改如下：

```
⋮
switch(score/10)                /*将区间值转换为整型数据*/
{
    case 10:
    case 9: grade='A';break;    /*90 分以上*/
    case 8: grade='B';break;    /*80～89 分*/
    case 7: grade='C';break;
    case 6: grade='D';break;
    default:grade='E';          /*60 分以下*/
}
⋮
```

思考：当输入的百分制成绩为 90.5 时，程序怎么修改？

【同步练习】 编写程序，计算个人应缴的所得税 tax=rate*(salary−1500)。假如：

当 salary＜1500 时，rate=0;

当 1500≤salary＜3000 时，rate=5％;

当 3000≤salary＜4500 时，rate=10％;

当 4500≤salary＜6000 时，rate=15％;

当 6000≤salary 时，rate=20％。

提示：多种情况的判断，采用选择分支结构中的 switch 语句。其中，switch 语句中“表达式”要求为整型数据。

3.5 应 用 举 例

【例 3-10】 编程实现，输入一个字符，若该字符是一个大写英文字母，则直接输出；若该字符是一个小写英文字母，则将它转换为大写英文字母并输出；若该字符是一个数字字符，则将它转换为整数并输出；若为其他字符，则输出“当前字符为无效字符!”。

编程提示：判断字符为大写英文字母的条件为(ch>='A')&&(ch<='Z')，不需要任何转换；判断字符为小写英文字母的条件为(ch>='a')&&(ch<='z')，需要将小写英文字母转换为大写英文字母，由于大写英文字母与对应的小写英文字母的 ASCII 码相差 32，因此 ch=ch−32；判断字符为数字字符的条件为(ch>='0')&&(ch<='9')，需要将数字字符转换为整型数字，由于数字字符与对应整型数字的 ASCII 码相差 48，因此 ch=ch−'0'。

```
#include <stdio.h>
int main()
{
    char ch;
    printf("请输入一个字符：");
    scanf("%c",&ch);
    if (ch>='A'&&ch<='Z')
        printf("当前字符为大写英文字母\n");
    else if (ch>='a'&&ch<='z')
    {
        ch=ch-32;              /*大写英文字母与对应小写英文字母的 ASCII 码相差 32*/
        printf("当前字符为小写英文字母,转换为大写英文字母,即%c\n",ch);
    }
    else if (ch>='0'&&ch<='9')
    {
        ch=ch-'0';             /*数字字符与对应整型数字的 ASCII 码相差 48*/
        printf("当前字符为数字字符,转换为整数,即%d\n",ch);
    }
    else
        printf("当前字符为无效字符! \n");
    return 0;
}
```

程序运行结果如下：

```
请输入一个字符：a↙
当前字符为小写英文字母，转换为大写英文字母，即 A
```

【例 3-11】 编程实现，求 $ax^2+bx+c=0$ 的解。

编程提示：由键盘输入 a、b、c。假设 a、b、c 的值任意，方程有以下几种可能。

（1）$a=0$，这不是二元一次方程；

（2）$b^2-4ac=0$，有两个相同的实根；

（3）$b^2-4ac>0$，有两个不同的实根；

（4）$b^2-4ac<0$，有两个共轭复根。以 $p+qi$ 和 $p-qi$ 的形式输出复根。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    double a,b,c,disc,x1,x2,realpart,imagpart;
    printf("请输入 a,b,c 的值(以","间隔):");
    scanf("%lf,%lf,%lf",&a,&b,&c);          /*输入时以","间隔*/
    if (fabs(a)<=1e-6)                      /*此方程不是二元一次方程*/
    {
        printf("这不是二元一次方程!\n");
        exit(0);
    }
    else
    {
        disc=b*b-4*a*c;
        if (fabs(disc)<=1e-6)               /*方程有两个相同的实根*/
            printf("此方程有两个相同的实根：\nx1=x2=%8.2f\n",-b/(2*a));
        else
            if (disc>1e-6)                  /*方程有两个不同的实根*/
            {
                x1=(-b+sqrt(disc))/(2*a);
                x2=(-b-sqrt(disc))/(2*a);
                printf("此方程有两个不同的实根：\nx1=%8.2f\n x2=%8.2f\n",x1,x2);
            }
            else                            /*方程有两个共轭复根*/
            {
                realpart=-b/(2*a);          /*realpart 是复根的实部*/
                imagpart=sqrt(-disc)/(2*a); /*imagpart 是复根的虚部*/
                printf("此方程有两个共轭复根：\n");
                printf("x1=%8.2f+%8.2fi\n",realpart,imagpart);
                printf("x2=%8.2f-%8.2fi\n",realpart,imagpart);
            }
    }
    return 0;
}
```

程序运行结果如下：

```
请输入 a,b,c 的值(以","间隔)：1,2,1↙
此方程有两个相同的实根：
x1=x2=   -1.00
```

或

```
请输入 a,b,c 的值(以","间隔)：1,5,6↙
```

```
此方程有两个不同的实根：
x1=   -2.00
x2=   -3.00
```

或

```
请输入 a,b,c 的值(以","间隔)：1,1,2↙
此方程有两个共轭复根：
x1=   -0.50+   1.32i
x2=   -0.50+   1.32i
```

程序中用 disc 代表 b^2-4ac，先计算 disc 的值，以减少以后的重复性计算。当判断 disc 是否等于 0 时，由于 disc 的值为实数，因此不能直接将 disc 与 0 比较，即 if(disc==0)。原因在于，实数在内存中是以浮点形式存储的，而浮点数并不是真正意义上的实数，只是其在某种范围内的近似。因此，只能用近似的方法将实数与 0 进行比较。

说明：fabs(*x*)是库函数，其功能是求 *x* 的绝对值。在调用此函数前需要在 main()函数前加#include <math.h>。

【例 3-12】 编程实现，输入年、月、日，判断当天是这一年中的第几天。

编程提示：正常年份每个月中的天数是已知的，但 2 月比较特殊，要判断输入的年份是否为闰年，若为闰年且月份在 3 月或 3 月以后时，总天数要再加 1。闰年的条件为年份能被 4 整除但不能被 100 整除或能被 400 整除。

```
#include <stdio.h>
int main()
{
    int day,month,year,sum,leap;
    printf("输入年、月、日(以空格间隔)：");
    scanf("%d%d%d",&year,&month,&day);          /*输入年月日,以空格间隔*/
    switch(month)                               /*判断输入月份的总天数*/
    {
        case 1:sum=0;break;
        case 2:sum=31;break;
        case 3:sum=59;break;
        case 4:sum=90;break;
        case 5:sum=120;break;
        case 6:sum=151;break;
        case 7:sum=181;break;
        case 8:sum=212;break;
        case 9:sum=243;break;
        case 10:sum=273;break;
        case 11:sum=304;break;
        case 12:sum=334;break;
    }
    sum=sum+day;
    if (year%4==0&&year%100!=0||year%400==0)    /*判断是否为闰年*/
        leap=1;
```

```
    else
        leap=0;
    if (leap==1&&month>=3)          /*若为闰年,且月份大于或等于 3,则天数加 1*/
        sum++;
    printf("当天是这一年中的第%d 天。\n",sum);
   return 0;
}
```

程序运行结果如下：

```
输入年、月、日(以空格间隔)：2008 8 8↙
当天是这一年中的第 221 天。
```

或改为

```
…
if ((year%4==0&&year%100!=0||year%400==0)&&month>=3)
    sum++;                          /* 若为闰年,且月份大于或等于 3,则天数加 1 */
…
```

【例 3-13】 编程实现，输入三角形的 3 条边 *a*、*b*、*c*，判断它们能否构成三角形。若能构成三角形，判断其为等腰三角形、等边三角形、直角三角形，还是一般三角形。

编程提示：为了区分特殊三角形和一般三角形，程序中定义了一个标志位标量 flag，flag 为 1 表示该三角形为一般三角形，flag 为 0 表示该三角形为特殊三角形。

本题涉及实数比较问题，因为实数运算的结果是有精度限制的，如按照勾股定理，先计算两个直角边的平方和，再计算其平方后得到的斜边，发现只能是一个近似值，因此不能使用

```
if (a*a+b*b==c*c||a*a+c*c==b*b||c*c+b*b==a*a)
```

直接判断经计算得到的两个实数是否相等，应使用下列方法：

```
if (fabs(a-b)<=1e-1||fabs(b-c)<=1e-1||fabs(c-a)<=1e-1)
```

如果精度要求不高，比较时用 1e−1 就可以。

```
#include <stdio.h>
#include <math.h>
int main()
{
    float a,b,c;
    int flag=1;                     /*置标志变量 flag 为非 0 值*/
    printf("输入 a,b,c(以空格间隔)：");
    scanf("%f%f%f", &a, &b, &c);
    if (a+b>c&&b+c>a&&a+c>b)         /*如果满足三角形的条件*/
    {
        printf("这是一个");
        if (fabs(a-b)<=1e-1&&fabs(b-c)<=1e-1)
```

```
        {
            printf("等边");           /*等边*/
            flag=0;                   /*置标志变量 flag 为 0 值*/
        }
        else if (fabs(a-b)<=1e-1||fabs(b-c)<=1e-1||fabs(c-a)<=1e-1)
        {
            printf("等腰");           /*等腰三角形*/
            flag=0;                   /*置标志变量 flag 为 0 值*/
        }
        else if (fabs(a*a+b*b-c*c)<=1e-1||fabs(a*a+c*c-b*b)<=1e-1||
             fabs(c*c+b*b-a*a)<=1e-1)
        {
            printf("直角");           /*直角三角形*/
            flag=0;                   /*置标志变量 flag 为 0 值*/
        }
        if (flag)                     /*若标志变量 flag 非 0,则是一般三角形*/
            printf("一般");
        printf("三角形! \n");
    }
    else                              /*不满足三角形的条件*/
        printf("这不是三角形! \n");
        return 0;
}
```

程序运行结果如下：

```
输入 a,b,c(以空格间隔)：4 4 5↙
这是一个等腰三角形!
```

或

```
输入 a,b,c(以空格间隔)：4 4 4↙
这是一个等边三角形!
```

或

```
输入 a,b,c(以空格间隔)：3 4 5↙
这是一个直角三角形!
```

或

```
输入 a,b,c(以空格间隔)：3 4 6↙
这是一个一般三角形!
```

或

```
输入 a,b,c(以空格间隔)：3 4 9↙
这不是一个三角形!
```